**Cuprins:**

[**Introducere** 2](#_Toc100612439)

[**1.** **Bazele logice a calculatoarelor numerice Noţiuni de bază ale algebrei logice (Booleene).** 3](#_Toc100612440)

[**1.1.** **Bazele aritmetice ale calculatoarelor numerice Sisteme de numeraţie și coduri** 3](#_Toc100612441)

[**1.1.1.** **Conversia numerilor dintr-un SN în altul** 3](#_Toc100612442)

[**1.1.2.** **Convertirea numărului în octal** 5](#_Toc100612443)

[**1.1.3.** **Sistem de enumerație hexazecimal** 7](#_Toc100612444)

[**1.1.4.** **Codurile binar – zecimale BCD** 8](#_Toc100612445)

[**1.1.5.** **Coduri alfanumerice** 9](#_Toc100612446)

[**1.2.** **Minimizarea funcţiilor booleene** 9](#_Toc100612447)

[**1.2.1.** **Metoda diagramelor Karnaugh** 10](#_Toc100612448)

[**1.2.2.** **Metoda Quine-McCluskey** 11](#_Toc100612449)

[**1.2.3.** **Formele grafice de reprezentare a funcțiilor logice Tabelul de adevăr:** 12](#_Toc100612450)

[**1.2.4.** **Formele algebrice de reprezentare a funcțiilor logice** 13](#_Toc100612451)

[**1.3.** **Sinteza circuitelor secvenţiale:** 14](#_Toc100612452)

[**1.3.1.** **Bistabilele Circuitele basculante** 15](#_Toc100612453)

[**1.3.2.** **Registrele** 18](#_Toc100612454)

[**1.3.3.** **Numărătoarele** 19](#_Toc100612455)

[**2.** **Proectarea convertorului de cod** 21](#_Toc100612456)

[**2.1.** **Minimizarea FDNP și FCNP.** 21](#_Toc100612457)

[**2.1.1.** **Minimizarea funcțiilor folosint metoda lui Karno: FDNP** 22](#_Toc100612458)

[**2.1.2.** **Minimizarea funcțiilor folosint metoda lui Karno: FCNP** 25](#_Toc100612459)

[**2.2.** **Construirea și simularea convertorului, efectuaţi sinteze-i unui binar-zecimal pe 7 segmente** 27](#_Toc100612460)

[**2.3.** **Circuit login secvential (Numaroator asincron)** 31](#_Toc100612461)

[**2.4.** **Transformările binare.** 33](#_Toc100612462)

[**Concluzie:** 35](#_Toc100612463)

[**Bibliografie:** 36](#_Toc100612464)

# **Introducere**

Dispozitivele numerice (digitale) sunt componentele de bază ale calculatoarelor electronice şi ale altor sisteme şi aparate destinate procesării informaţiei. Aceste structuri numerice, pe care se bazează o foarte mare parte a tehnologiilor moderne din domeniul electronicii, asigură precizia, fiabilitatea, viteza şi complexitatea necesare prelucrării datelor. Începînd cu structurile logice cele mai simple şi pînă la circuitele secvenţiale complexe sau logica programată se utilizează pe scară largă diferite metode şi tehnici de proiectare (sinteză) şi analiză a acestora. Pentru a utiliza cu succes aceste metode şi tehnici proiectantul dispozitivelor numerice trebuie să aibă pregătirea teoretică respectivă şi să dispună de deprinderile practice necesare.

Dispozitiv numeric – orice componentă a calculatorului care poate fi descrisă cu ajutorul logicii algebrei Booleene. Sinteza dispozitivului numeric - este procesul de elaborarea structurii dispozitivului numeric începând cu descrierea destinaţiei dispozitivului respectiv şi terminînd cu schema definitivă a acestuia.

Analiza - analiza dispozitivului numeric reprezintă procedura de descriere a funcţionării dispozitivului numeric respectiv şi de descriere formală a lui în cazul când schema acestui dispozitiv deja există. Acest domeniu este destinată studierii metodelor de sinteza elementelor funcţionale, a automatelor numerice, de comandă şi operaţonale care stau la baza oricărui calculator numeric.

Algebra logică sau algebra booleană este ştiinţa care studiază propoziţiile. Prin noţiunea da propoziţie se înţelege o expresie care comunică o informaţie şi această informaţie poate avea doar două sensuri: adevărat sau fals(1 şi 0). Aceste stări le poate avea orice dispozitiv tehnic.

Sistemul Logic Works este destinat proiectării şi simulării circuitelor digitale. Asamblarea circuitelor se efectuează cu ajutorul elementelor logice şi circuitelor combinaţionale şi secvenţiale care se conţin în bibliotecile sistemului. Circuitul asamblat se analizează cu ajutorul diagramelor de timp şi a elementelor de vizualizare a valorilor logice generate de circuit.

# **Bazele logice a calculatoarelor numerice Noţiuni de bază ale algebrei logice (Booleene).**

Variabila logică (booleeană) este o variabilă care poate avea doar două valori 0 şi 1. Pentru 1 se subînţelege că această variabilă reprezintă un adevăr iar pentru 0 – reprezintă o eroare.

**Funcţia logică** este o funcţie dependentă doar de variabilele logice şi care poate avea numai valori de 0 şi 1. Dependenţa valorii funcţiei de valorile variabilelor este determinată de operaţii logice îndeplinite în funcţia respectivă pentru a afla valoarea acesteia. Există mai multe tipuri de funcţii booleene.

**Poartă logică** este un dispozitiv electronic numeric elementar care implementează o funcțiune logică elementară, numărul de intrări al căruia este egal cu numărul variabilelor funcției și o singură ieșire. Porțile logice sunt structurile de bază care permit realizarea unor funcții logice complexe în circuitele integrate digitale.

## **Bazele aritmetice ale calculatoarelor numerice Sisteme de numeraţie și coduri**

Sistem de numeraţie se numeşte complexul de reguli şi simboluri cu ajutorul cărora se reprezintă numerele. Simbolurile prin care se reprezintă informaţia numerica se numesc cifre. Numărul de cifre prin care sunt reprezentate numerele se numeşte baza sistemului de enumerare.

### **Conversia numerilor dintr-un SN în altul**

Regulile de conversie a numerelor dintr-un sistem de numeraţie în altul sunt diferite pentru partea întreaga şi partea fracţionară. Trebuie sa efectuam conversii din r -->h Ca sa stabilim aceste reguli, considerăm numărul (y) că va avea n+1 cifre în baza noului sistem de numerație h va fi:

Y=bn\*hn+n-1\*hn-1…b1\*h1+b0\*h0

Împărţim şi stânga şi dreapta la baza S.N h:

Y/h=bn\*hn-1+bn-1h n-2…b1+b0/h

Notam: bn\*hn-1+bn-1h n-2…b1

prin y1 Acum înmulţim cu h Rezultatul va fi:Y=y1\*h+b0 Y1=y2\*h+b1

**Regula de conversie:**

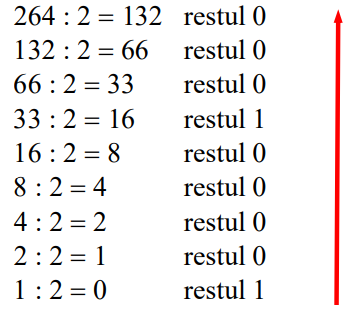
Pentru a efectua conversia numerelor întregi numărul iniţial reprezentat în baza sistemului de numerație (r) şi câturile obţinute ulterior se împart succesiv la baza S.N (h) (în care se efectuează conversia).

Împărţirile se fac până când, cîtul obţinut devine mai mic ca (h). Cifrele obţinute în noul sistem de numerație sunt (=cu) resturile de la împărţire în ordinea inversa a obţinerii lor. Cea mai semnificativa cifra a num.(în noul S.N) este egală cu ultimul cît.

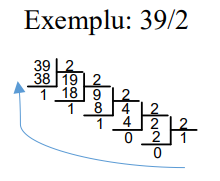
Conversia din zecimal în binar se face conform următoarelor reguli:

* Numărul zecimal se împarte la baza noului S. N. în urma împărțirii obținem cîtul întreg și restul împărțirii;
* Dacă cîtul obținut este diferit de 0 ne întoarcem la punctul 1, cîtul obținut la etapa precedentă îl folosim în calitate de deîmpărțit la etapa următoare; 3. Dacă cîtul obținut este egal cu 0, toate resturile obținute se folosec pentru reprezentarea numărului în S. N nou, în ordinea inversă a obținerii acestar resturi.

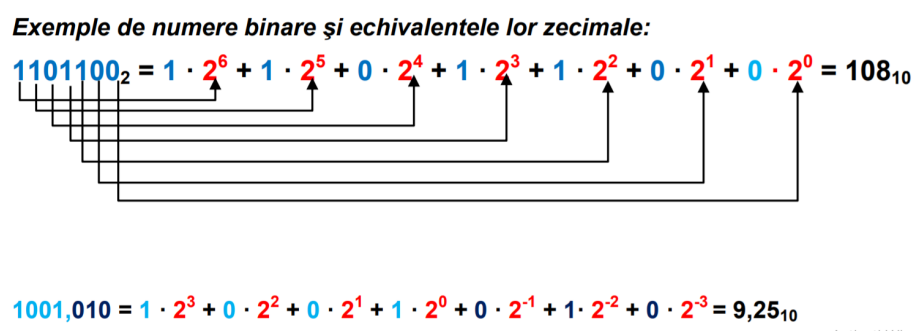
Exepmlu: Să convertim numărul 264 din baza 10 în baza 2. Facem împărțiri consecutive



Deci, 264(10)=100001000(2). Verificarea a facem conform formulei: 100001000(2)=1⋅2 8+1⋅2 3=264(10).

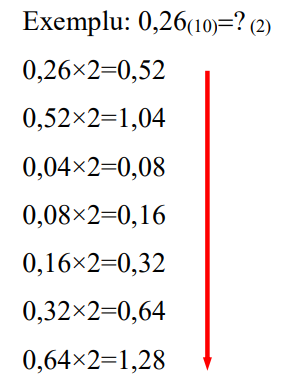


3910=1001112 1\*25+0\*24+0\*23+1\*22+1\*21+1\*20=32+0+0+4+2+1=39



Regula de conversie: numărului iniţial reprezentat în baza sistemului de numeraţie (r) şi produsele intermediare obţinute ulterior se înmulţesc succesiv la baza SN (h) numărul de înmulţiri se efectuează până se obţine numărul necesar de cifre după virgulă pentru a avea exactitatea dorita. Dacă trebuie obţinută aceeaşi exactitate a numărului în ambele sisteme de numeraţie atunci, daca în sistemul de numeraţie cu baza mai mica (în cazul nostru este r) avem (n) cifre, iar în sistemul de numeraţie cu baza mai mare avem (m) cifre atunci relaţia dintre n şi m este n=k\*m unde k=logrh

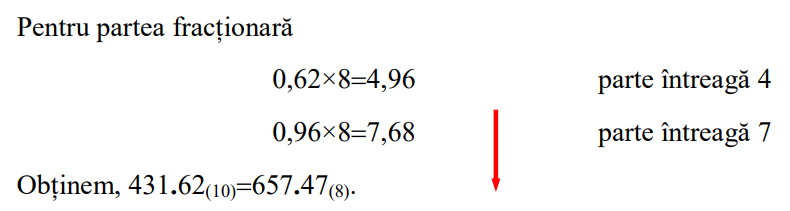
Deci pentru conversia numărului fracționar din baza 10 în altă partea fracționară a numărului se înmulțește consecutiv la baza noului sistem de enumerație cifrele părții întregi vor reprezenta cifrele noului sistem de enumerație, acestea se vor înscrie în ordinea apariției acestora.



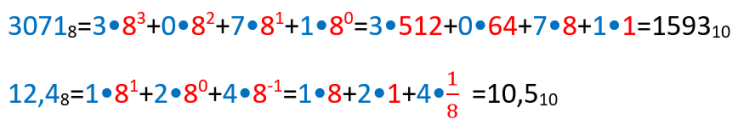
### **Convertirea numărului în octal**

Pentru conversia numîrului din S. N. zecimal în S. N. octal împărțim succesiv la baza S. N. (8), pînă nu obținem cifra cîtului < 8, resturile obținute în urma împărțirii se înscriu în ordinea inversă obținerii lor (pentru parte întregă), pentru conversia părții fracționare se efectuează înmulțiri consecutive la 8 (înmulțirile se fac pînă nu obținem precizia numărului dorită sau 0-după virgulă), cifrele rezultatului vor reprezenta cifrele întregi înscrise în ordinea obținerii acestora. Exemplu: avem numărul zecimal 431.62. Facem conversia:

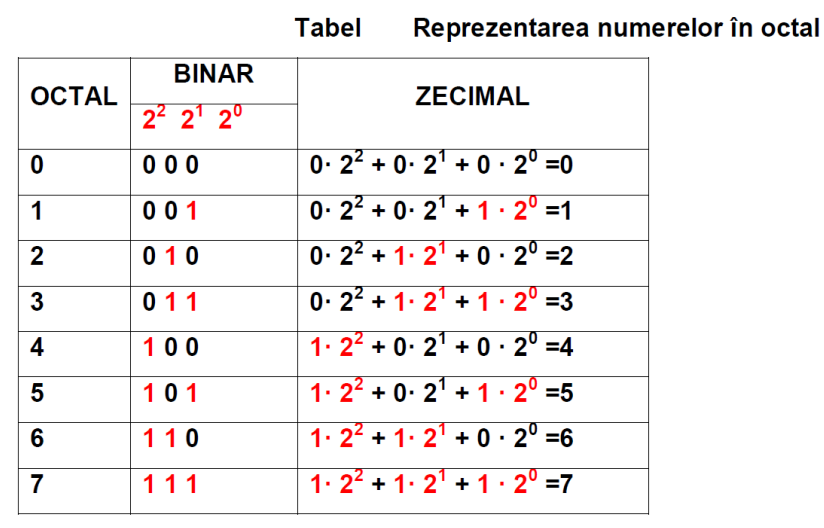




Dacă r=8 se obţine S.N. octal, adică aceast sistem de numeraței utilizează 8 caractere (cifre). S={0, 1, 2, 3, 4, 5, 6, 7} Exemple de numere octale și echivaletele lor zecimale:



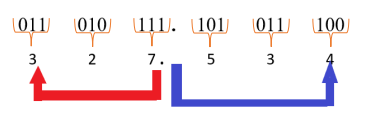
Fiecărui caracter din S.N. octal îi corespunde un echivalent (șir) de 3 biți în binar deoarece 8 este o pondere a lui 2 (log28=3) după cum este reprezentat în tabelul de mai jos



Regula de conversie numărului binar în număr octal :

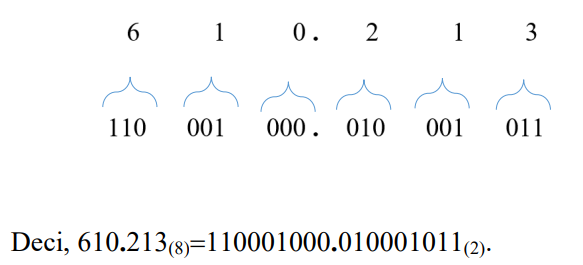
* Cifrele binare (pentru partea întreagă de la virgulă spre stînga, pentru cea fracționară de la virgulă spre dreapta) se grupează a câte 3 formînd triade ;
* Fiecare triadă se înlocuește cu cufra octală conform tabelului de mai sus

Exemplu: Fie avem numărul: 11010111.1010111 Împărțim numărul în triade, pe cere ulterior le înlocuim cu echuvalentul cifrei conform tabelului de mai sus, și obținem:



Deci 11010111.1010111(2)=327.534(8). În acest exemplu triadele din extrema dreapta și stînga au fost complementate cu zerouri.

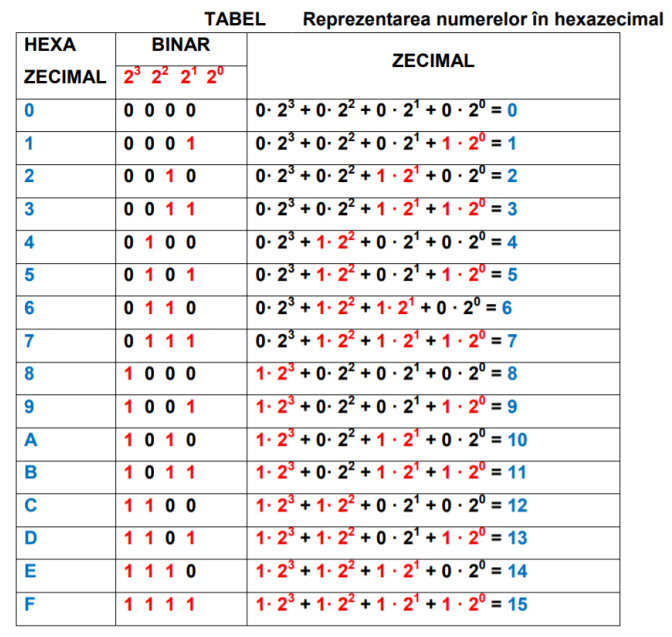
Conversia inversă (din octal în binar) constă în înlocuire cifrei octale cu o triadă (trei cifre) binare conform tabelului de mai sus. De exemplu: Avem numărul N(8)=610.213 :

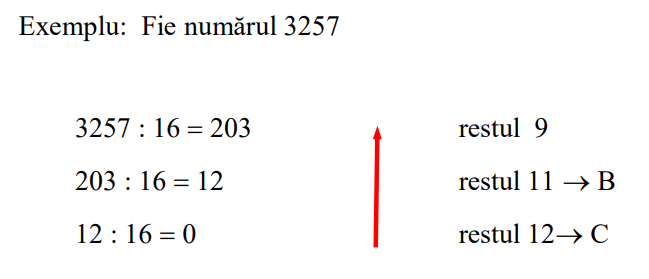


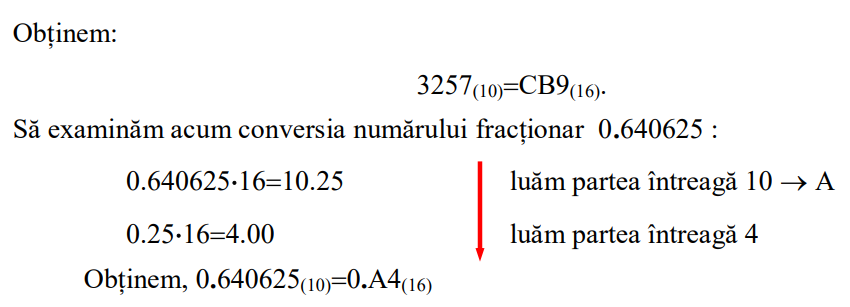
### **Sistem de enumerație hexazecimal**

Dacă r=16 se obţine S.N. hexazecimal, adică aceast sistem de numeraței utilizează 16 caractere (cifre). S={0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}. Conversia din S. N. binar în hexazecimal are loc analogic cu conversia din binar în octal cu următoarea diferență (deoarece log216=4) cifrele din reprezentarea binară se grupează a cîte 4 (tetradă) la fel ca și în caul precedent și se înlocuesc cu echivalentul lor hexazecimal, conversia inversă (din hexazecimal în binar) le fel este valabilă. Cifrele hexazecimale și echivalenții lor binari sunt aduse în tabelul de mai jos.

Conversia numărululi zecimal în hezazecimal are loc prin împărțiri consecutive la baza sistemului (16) – pentru parte întreagă, rezultetul se va scrie în ordinea înversă a obținerii lor și înlocuind cifrele > 9 și <16 cu literele respective (vezi tabelul de mai jos) după cum este arătat în exemplul următor.







### **Codurile binar – zecimale BCD**

Codurile binar- zecimale În afara sistemelor de numerație expuse mai sus, dispozitivele numerice operează și cu alte reprezentări numerice. Drept exemplu servesc codurile binar – zecimale BCD (Binary Coded Decimal). Calculatoarele moderne au şi regimul de funcţionare în sistemul de numeraţii zecimale.

Acest regim se foloseşte în codurile când problemele care se rezolvă necesită prelucrarea a masivelor mari de cifre zecimale şi se confruntă cu încărcarea acestor masive în calculator şi extragerea masivelor de date în volum mare din calculator. În acest caz pentru a evita pierderile de timp pentru conversia datelor la intrarea şi ieşirea din calculator, datele se prelucrează în sistemul binar-zecimal.

Acestea sunt problemele de statistică economică etc. în acest caz fiecare cifră zecimală este reprezentată printr-un cod binar - zecimal de patru biţi sunt mai multe variante de codificare a cifrelor zecimale şi multitudinea codurilor utilizate în acest sens se împart în grupuri după următoarele criterii:

* I criteriu- este sau nu este codul ponderat

Definiţie:

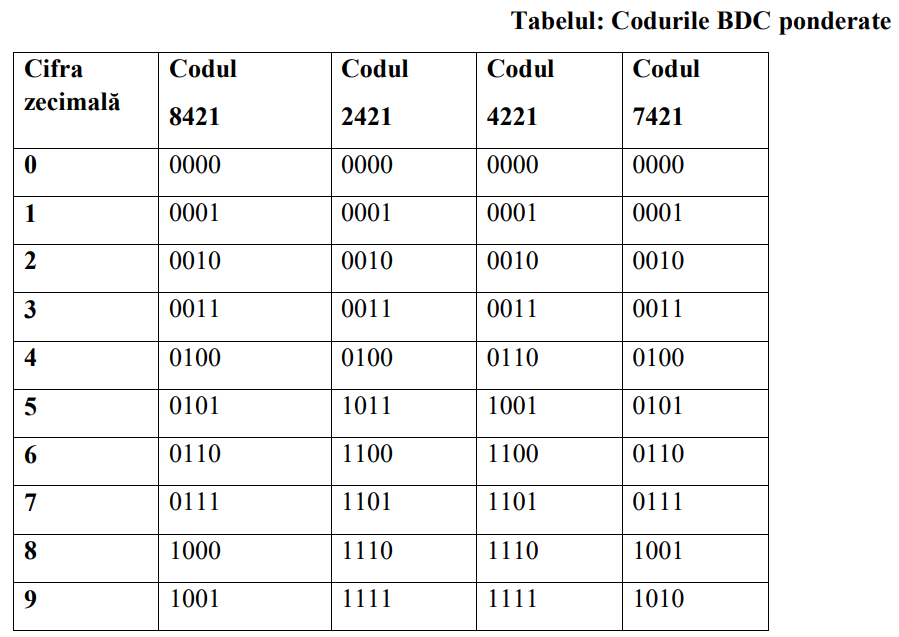
Codul se numeşte ponderat dacă pentru fiecare combinaţie de cifre binare care codifică cifrele zecimale α4 α3α2α1, care codifică cifrele zecimale, este valabil următoarea formulă: a= α1P1+α2P2+α3P3+α4P4 unde P1, P2, P3, P4 sunt valorile care reprezintă ponderea fiecărei poziţie a codului binar zecimal.

* II criteriu- criteriu de complementarism

Definiţie:

Un cod se numeşte complementar dacă se respectă următoarea condiţie pentru oricare cifră zecimală care în sumă ne dau cifra 9 codul numai dintre acest cifre se obţine inversa codului ceilalte cifre. Reieșind din cele expuse codurile binar zecimale pot fi ponderate și neponderate.

Coduri ponderate Un cod ponderat asociază fiecărei cifre zecimale o tetradă binară, ponderea fiecărui bit din tetradă fiind egală cu valoarea cifrei din denumirea codului În tabelul de mai jos sunt aduse echivalenții codurilor ponderate:



### **Coduri alfanumerice**

Informaţiile alfanumerice sunt cele care se reprezintă sub formă de text şi eventual conţin o grafică simplă bazată pe forme grafice predefinite de dimensiunea unui caracter. Cel mai utilizat sistem pentru reprezentarea informaţiilor alfanumerice este standardul ASCII (American Standard Coding for Information Interchange). Acest standard utilizează 7 biţi pentru reprezentarea codurilor alfanumerice (8 biţi în varianta extinsă). Sunt codificate următoarele tipuri de date:

* litere mari şi mici
* cifre zecimale
* semne de punctuaţie
* coduri de editare
* formatare a textului
* coduri de control al transferului de date

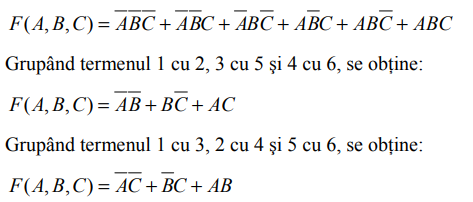
Alte coduri: EBCDIC Extended BCD Interchange code 8 biţi UNICODE 16 biţi Standardul "unicode" extinde codificarea ASCII prin adăugarea unui octet suplimentar. Astfel anumite litere speciale (litere greceşti, litere arabe, litere cu semne speciale, etc.) sau semne grafice sunt reprezentate pe 16 biţi.

## **Minimizarea funcţiilor booleene**

Minimizarea constă în obţinerea formei celei mai simple de exprimare a funcţiilor booleene în scopul reducerii numărului de circuite şi a numărului de intrări ale acestora.

Metoda algebrică Metoda algebrică constă în aplicarea succesivă a postulatelor şi teoremelor algebrei booleene scrise sub formă canonică disjunctivă sau conjunctivă. O funcţie care nu este specificată iniţial sub o formă canonică poate fi adusă la această formă.

În vederea minimizării, se urmăreşte reducerea numărului de termeni ai expresiei, a numărului de apariţii ale variabilelor şi a numărului de variabile din fiecare termen. Apariţia unei variabile complementate sau necomplementate reprezintă un literal. Considerăm următoarea funcţie:



Acestea reprezintă expresii minimale ale funcţiei. Deci, o expresie minimală a unei funcţii nu este, în mod obligatoriu, unică. Metoda algebrică necesită experienţă, devenind dificilă dacă expresia iniţială a funcţiei este complicată. Un alt dezavantaj este faptul că nu se poate stabili cu uşurinţă dacă forma obţinută este minimă sau se mai poate simplifica. Din aceste motive, în practică se utilizează metode grafice de minimizare.

### **Metoda diagramelor Karnaugh**

Metoda diagramelor Karnaugh Folosirea unei diagrame pentru simplificarea funcţiilor booleene a fost sugerată pentru prima dată de E. Veitch. Ulterior, M. Karnaugh propune de asemenea o formă de diagramă în acelaşi scop, rezultând diagrama Karnaugh. Această diagramă se utilizează în mod curent pentru reprezentarea funcţiilor booleene cu un număr relativ mic de variabile.

Reprezentarea funcţiilor prin diagrama Karnaugh O diagramă Karnaugh constituie o variantă modificată a unui tabel de adevăr. În general, o diagramă Karnaugh pentru o funcţie booleană de n variabile se reprezintă sub forma unui pătrat sau dreptunghi împărţit în 2n pătrate (compartimente), fiecare pătrat fiind rezervat unui termen canonic al funcţiei.

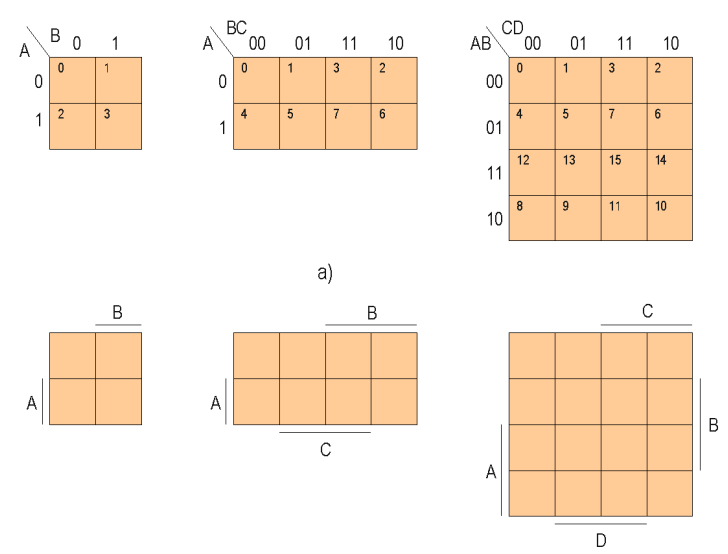


Figura 1. Diagrame Karnaugh pentru funcţiile booleene de 2, 3 şi 4 variabile.

O diagramă Karnaugh se notează fie indicând pe linie şi coloană combinaţiile corespunzătoare fiecărui pătrat şi ordinea variabilelor, fie indicând domeniul fiecărei variabile.

Pentru a se putea reprezenta în mod simplu funcţii date în mod convenţional prin indicii termenilor canonici, se poate nota fiecare compartiment cu indicele termenului canonic corespunzător. O diagramă Karnaugh este astfel organizată încât două pătrate vecine (cu o latură comună) pe o linie sau pe o coloană corespund la combinaţii care diferă printr-o singură cifră binară, deci la doi termeni canonici care diferă printr-o singură variabilă, care apare într-unul din termeni sub formă complementată, iar în celălalt sub formă necomplementată. Asemenea două pătrate veci

Se consideră adiacente şi pătratele aflate la capetele opuse ale unei linii, respectiv coloane. De aceea, este convenabil să se privească aceste diagrame ca suprafeţe care se închid la margini. De exemplu, la o diagramă de 4 variabile, pătratele 0 şi 2, sau 0 şi 8 sunt adiacente. Deoarece unui termen canonic cu n literale îi corespund n termeni care diferă printr-un literal, într-o diagramă cu n variabile fiecare pătrat are n pătrate adiacente.

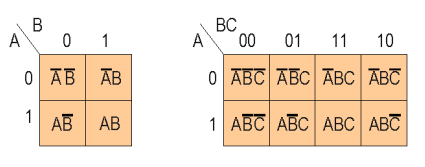


Figura 2. diagramă de 4 variabile

În general, o funcţie booleană de n variabile se poate reprezenta în spaţiul ndimensional al celor n variabile sub forma unui hipercub. Fiecărui termen canonic al funcţiei îi corespunde un vârf al hipercubului. Un grup de 2m puncte (m < n), fiecare dintre ele adiacente la m puncte ale grupului, se numeşte subcub, şi se spune că subcubul acoperă aceste puncte ale grupului.

În diagrama Karnaugh, fiecare pătrat corespunde unui vârf al cubului ndimensional din reprezentarea geometrică a funcţiei. O funcţie booleană dată sub forma canonică disjunctivă poate fi reprezentată pe o diagramă Karnaugh marcând cu 1 pătratele corespunzătoare mintermenilor funcţiei.

### **Metoda Quine-McCluskey**

Metoda Quine-McCluskey (sau metoda primilor implicați ) este un [algoritm](https://koaha.org/wiki/Algoritmo) dezvoltat de [Willard Van Orman Quine](https://koaha.org/wiki/Willard_Van_Orman_Quine) și [Edward McCluskey](https://koaha.org/wiki/Edward_McCluskey) care este utilizat în [rețele](https://koaha.org/wiki/Rete_combinatoria) logice [combinaționale pe](https://koaha.org/wiki/Rete_combinatoria) două niveluri pentru minimizarea unei funcții booleene a *n* [variabile](https://koaha.org/wiki/Variabile_(matematica)) .

Metoda este funcțională identică cu [harta Karnaugh](https://koaha.org/wiki/Mappa_di_Karnaugh) , dar forma sa tabelară o face mai eficientă pentru crearea computerului; oferă, de asemenea, un mod determinist de a testa minimizarea unei funcții booleene.

Metoda constă din doi pași:

* Identificați toți [implicații principali](https://koaha.org/wiki/Implicante) ai funcției.
* Întoarceți implicații principali găsiți într-un tabel pentru a obține implicații primi esențiali ai funcției.

**Complexitate**

Deși este mai practic decât hărțile Karnaugh pentru funcții cu mai mult de 4 variabile, metoda Quine-McCluskey are încă o gamă limitată de utilizare, deoarece problema rezolvată de algoritm ( [satisfacția booleană](https://koaha.org/wiki/Soddisfacibilit%C3%A0_booleana) ) este [NP-dificilă](https://koaha.org/wiki/NP-difficile) : timpul de [rulare](https://koaha.org/wiki/Runtime) crește [exponențial pe măsură](https://koaha.org/wiki/Funzione_esponenziale) ce numărul de intrări crește.

Se poate arăta că pentru o funcție de *n* variabile limita superioară a numărului de implicați primi este de 3 *n* / *n* . Dacă *n* = 32 pot exista mai mult de 6,5 \* 10 15 implicați primi. Prin urmare, funcțiile cu un număr mare de variabile booleene trebuie reduse la minimum cu metode euristice, cum ar fi Espresso minimizator logic euristic.

**Metodă:**

Metoda constă din două faze principale: căutarea primilor implicați și căutarea ulterioară a acoperirii optime. Considerăm minimizarea sub forma sumei de produse (numită și SOP, din acronimul englez *sum of products* ), dar totul este ușor extensibil la forma de produs de sume (sau POS, *produs de sume* ).

În prima fază, simplificarea tipului se aplică sistematic.

Adică proprietatea distributivă a produsului în raport cu suma, unde P indică orice termen al produsului (mintermine). Evident, metoda poate fi extinsă și la funcții care nu sunt complet specificate și la circuite cu mai multe ieșiri.

Prima fază constă din pașii:

* pentru a tabela toate mintermele funcției în formă binară, în ordine crescătoare ca și pentru tabelul adevărului;
* comparați toți termenii în mod exhaustiv: termenii care diferă cu un singur bit sunt simplificați (distanță de Hamming unitară) și marcați, deoarece au contribuit la crearea unui implicant;
* apoi creați un nou tabel cu toți termenii de produs marcați care ies din primul tabel și repetați pasul 2);
* procesul se încheie atunci când nu se mai pot face reduceri.

În momentul construirii tabelului este ușor de văzut că nu trebuie neapărat să comparăm toți termenii între ei, ci de fapt numai acei termeni adiacenți care diferă doar cu un bit 1. Apoi grupăm în tabel termenii care au un număr egal de 1 în minterm.

### **Formele grafice de reprezentare a funcțiilor logice Tabelul de adevăr:**

Tabelul de adevăr reprezintă lista valorilor funcţiei logice pentru fiecare combinaţie a variabililor de intrare sau pentru fiecare cuvânt de intrare. Un tabel de adevăr care reprezintă m funcţii de n variabile fiecare va avea următoarele dimensiuni n + m – coloane şi 2n – rânduri.

Unde n este numărul de variabile iar m numărul de funcţii. În acest tabel de adevăr sînt incluse toate combinaţiile posibile care le pot avea variabilele. În fiecare rând este reprezentată una din combinaţiile posibile a variabilelor de intrare şi valorile tutror funcţiilor pentru respectiva combinaţie. Primele n coloane a tabelului de adevăr reprezintă valorile variabilelor funcţiilor câte o coloană pentru fiecare variabilă iar următoarele m coloane reprezintă valorile funcţiei respective pentru fiecare variabilă.

**Schema logică**

În schema logică fiecare operaţie logică este înlocuită printr-un echivalentă aşa numită poartă logică care prin diferite mijloace tehnice poate materializa funcţie logică. Poartă logică – este reprezentarea tehnică (materială) a unei operaţii logice sau cu alte cuvinte elementul logic realizează una din operaţiile logice având ca bază de realizare procese electronice, electrice sau mecanice. Fiecare operaţie logică are un semn convenţional care se realizează cu ajutorul porțilorlor logice.

**Costul schemei logice** reprezintă totalitatea intrărilor în porți logice care realizează o funcție logică. Costul se notează cu C și se măsoară în unități convenționale λ.

**Timpul de reșinere** a schemei este numărul maxim de porți logice prin care trece semnalul de la intrare spre ieșire. Se notează cu Td și se măsoară în τ.

Deci pentru schema de mai sus avem C=14 λ Td =2 τ.

**Diagrama de timp** – se folosește pentru analiza dependenței funcțieii logice de variabile de intrare în timp. La fel se folosește pentru studierea riscurilor dinamice și statistice în sistem. Risc este un semnal fals care poate duce la erori în lucrul sistemelor.

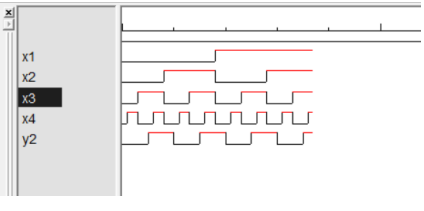


Figura 3. Diagrama de timp.

### **Formele algebrice de reprezentare a funcțiilor logice**

Forma analitică de reprezentare a funcţiilor logice presupune reprezentarea funcţiei prin operaţii logice (conjuncţie, disjuncţie şi negări). Asemenea forme sînt mai multe:

Conjuncţia (operaţia logică ŞI) x 1 x 2 ...x n se numeşte elementară dacă fiecare variabilă care intră în compoziţia ei este scrisă o singură dată. Rangul acestei conjuncţii este egal cu numărul de variabile din această conjuncţie.

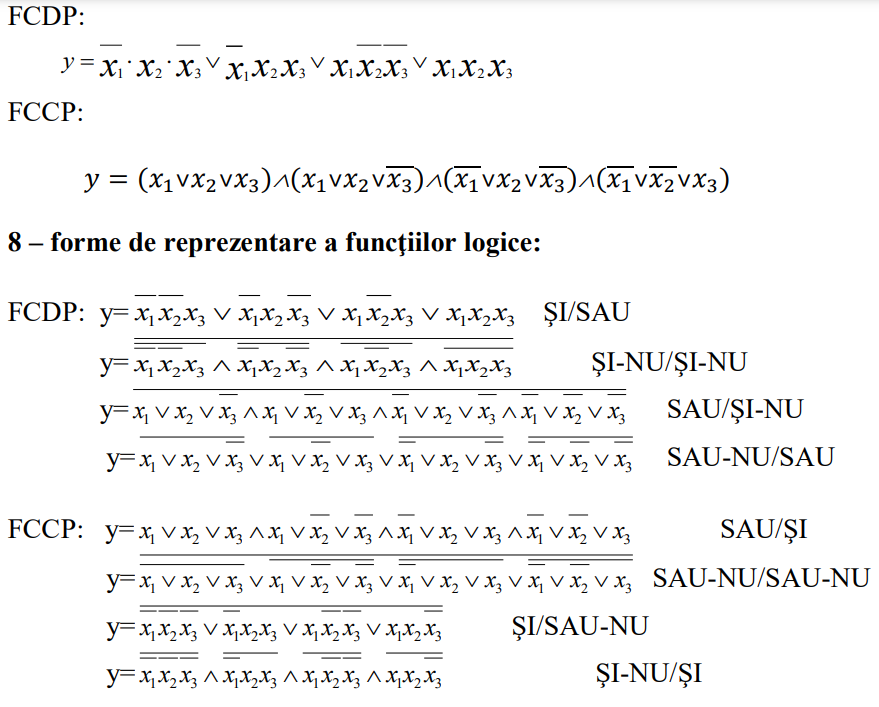
Disjuncţia (operaţia logică SAU) x 1 +x 2 +...+x n se numeşte elementară dacă fiecare variabilă care intră în compoziţia ei este scrisă o singură dată.

Disjuncţia conjuncţiilor elementare de rangul n a unei funcţii care are n variabile se numeşte

**Forma Canonică Disjunctivă Perfectă FCDP.**

Regula de formare a formei: pentru fiecare combinaţie de variabile unde funcţia este egală cu 1 se înscrie conjuncţia formată din variabilele acestei funcţii ţinînd seama de următoarea regulă: dacă variabila intră în combinaţie cu valoarea 1 atunci ea se înscrie fără schimbări în caz contrar ea se inversează (este scrisă cu negație). Toate conjuncţiile se unesc prin disjuncţii.

**Forma Canonică Conjunctivă Perfectă FCCP** – reprezintă conjuncţia disjuncţiilor elementare. Regula de formare a formei: pentru fiecare combinaţie de variabile unde funcţia este egală cu 0 se înscrie disjuncţia formată din variabilele acestei funcţii ţinînd seama de următoarea regulă: dacă variabila intră în combinaţie cu valoarea 0 atunci ea se înscrie fără schimbări, în caz contrar ea se inversează (este scrisă cu negație). Toate disjuncţiile se unesc prin conjuncţii.



## **Sinteza circuitelor secvenţiale:**

Circuitele logice secvenţiale (CLS) se caracterizează prin faptul că în orice moment de timp vectorul de ieşire al circuitului depinde nu numai de semnalele de la intrare din acel moment ci şi de semnalele de la intrare aplicate în momentele de timp anterioare.

Această dependenţă se datorează prezenţei în CLS a unor elemente de memorie, care reprezintă mai multe stări logice stabile, comandate prin intrări şi participă la formarea semnalelor de ieşire. Deci un CLS conţine o schemă combinaţională completată cu o structură de memorare. Starea CLS este fixată cu ajutorul elementelor de memorare, numite circuite basculante bistabile, sau bistabile.

Circuitele secvenţiale memorează starea precedentă, în consecinţă la sinteza circuitelor logice secvenţiale trebuie de ţinut cont nu numai de datele de la intrare a circuitului ci şi de starea circuitelor în momentul dat. Starea CLS este determinată de totalitatea stărilor tuturor bistabilelor. Pentru un CLS cu n bistabile pot exista pânăla 2n stări. După modul de funcţionare (modul de transmitere a semnalelor) există 2 categorii principale de CLS:

* asincrone – comportarea este determinată de aplicarea pe intrări a semnalelor în momente oarecare; starea circuitului depinde de ordinea în care se schimbă semnalele;
* sincrone – comportarea este determinată de aplicarea pe intrări a semnalelor în momente discrete, bine determinate în timp; sincronizarea se realizează cu ajutorul unor impulsuri date de un generator de tact (ceas).

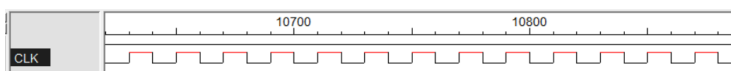


Figura 4. Ceas.

Sinteza unui CLS se efectuează în următoarele etape:

* Descrierea funcționării circuitul respectiv (prin text, desen, diagrame etc.);
* Reprezentarea acestei descrieri sub forma unui tabel de tranziţie;
* Deducerea funcţiilor logice şi minimizarea acestora;
* Implementarea funcţiilor minimizate sub forma unor reţele de comutare prin intermediul circuitelor integrate;

Tabelul de tranziţie se deosebeşte de tabelul de adevăr prin faptul că aici trebuie luat în consideraţie şi factorul timpului.

Acest tabel include următoarele părţi componente:

* valoarea variabilelor de intrare la orice moment de timp t;
* valoarea funcţiilor (vectorilor) de ieşire la momentul de timp t;
* valoarea funcţiilor (vectorilor) de ieşire, care vor rezulta la momentul de timp t+1 în urma aplicării semnalelor de intrare în momentul de timp precedent t.

Starea curentă a circuitului secvenţial se utilizează în calitate de variabilă, iar starea ulterioară în calitate de funcţie.

În calitate de valori a funcţiilor logice, care trebuie deduse şi minimizate, se iau valorile funcţiilor de ieşire la momentul t+1, iar în calitate de variabile a acestor funcţii se iau valorile variabilelor de intrare la momentul t şi valorile funcţiilor de ieşire la momentul t, adică toate datele care asigură tranziţia circuitului respectiv de la starea care o are în momentul t la starea care trebuie s-o aibă în momentul t+1.

### **Bistabilele Circuitele basculante**

***Bistabile*** (bistabile) sunt circuite logice secvenţiale care au două stări stabile distincte 0 și 1. Un bistabil poate păstra un timp nedefinit informaţia binară şi în acelaşi timp starea sa poate fi citită în orice moment. Bistabilul are 2 ieşiri, una directă și alta inversă.

La fel ca și orice CLS, bistabilele pot fi asincrone și sincrone.

Bistabilele asincrone comută (își schimbă starea) doar la aplicarea semnalelor de date, fără a avea intrarea de tact (CLK). Bistabilele sincrone comută doar atunci cand semanlul de tact este activ.

În dependență de modul de comutare, bistabilele se clasifică în: Latch (bistabile, care comută pe nivelul logic 0 sau nivelul logic 1 al semnalului de tact). Flip-flop (Bistabile master-slave care comută pe frontul pozitiv sau frontul negativ al semnalului de tact).

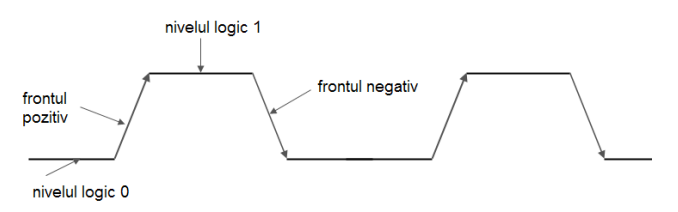


Figura 5. Flip-flop

**Cel mai simplu bistabil este RS (Reset-Set)**

T – indică că comutarea bistabilului se face întrun singur tact Bistabilul se afla în starea 1 atunci cînd iesirea Q=l, iar 𝑄 = 0, și se afla în starea 0 - cînd Q=0, iar 𝑄 = 1. Nici o data la ambele ieșiri nu pot avea valori egale. Aceste bistabile se numesc asincrone.

În calculatoarele numerice nu este convinabil de a folosi bistabile asincrone deoarece aceasta duce la pierderea informaţiei, acest neajuns se înlătură introducînd intrare de sinsronizare C sau “CLK”, acest semnal activează intrările S şi R numai cînd se află în starea logică 1, inhibîndu-le pe durata cît se află în stare logică 0. Când CLK=l ieșirile bistabilului vor depinde de S și R, pentru CLK=0 nu depinde de S și R. Acest tip de bistabil se numește sincron.

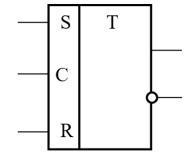


Figura 6. bistabila RS

**Bistabilul D**

Bistabilul D este realizatîn baza bistabilului RS respectînd condiția S = R , care exclude neajunsul bistabilului RS. Funcționarea bistabilului D poate fi descrisă în felul următor:

Qt+1=D

unde D – intrarea de date Bistabilul D este destinat pentru a memoriza cifra binara de la intrare pănă în momentul când la această intrare va fi о altă cifră. Schema acestui bistabil poate fi realizata în baza bistabilului RS prin conectarea intrărilor R și S printr-un inversor. Cînd CLK=0 avem regimul de pastrare.

Faptul că bistabilul D transmite datele la ieșire după cere intrările se blochează a primit denumirea de bistabilul latch – lacăt.

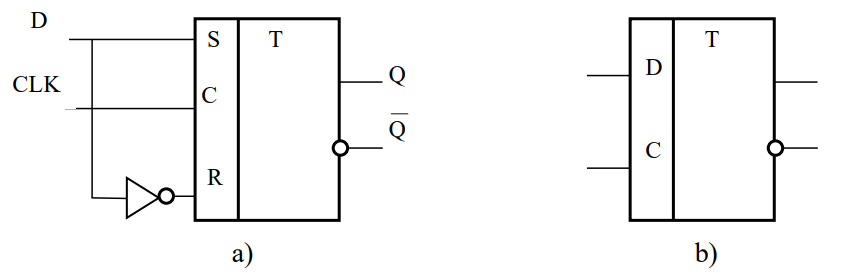


Figura 7. Bistabil D sincron: a – schema; b – simbolul grafic

**Bistabilul JK**

Necesitatea excluderii combinației interzise pentru R=S=1 a făcut necesitatea proiectării unui bistabil universal, care va funcționa pentru combinația 11 la intrarea acestuia.

Dacă la intrările JK este aplicat 1 bistabilul î-și va inversa starea. Funcția logică care descrie funcționarea bistabulului JK este:

* t KQ t JQ t Q = ∨ +1 Dacă J=K=0 î-și păstrează starea precedentă.

Dacă J=1 (și K=0) starea bistabilului va fi 1. Dacă K=1 (și J=0), starea bistabilului va fi 0. Dacă J=K=1 î-și inversează starea (0→1, 1→0). Adică J și K pot fi privite ca intrările S și R respectiv. Cu excepția stării J=K=1.

Comutarea bistabilului JK are lor pe frontul de cădere adică la trecerea din 1 în 0. JK– este un bistabil universal și are o aplicare foarte largă, poate funcționa în regim de bistabil RS (S se aplică la J și R la K cu exepția combinației J=l și K=l) care poate fi modificat prin aplicarea diferitor semnale ca sa functioneze în regim de bistabil D (datele se aplică pe linia J și intrările sunt legate cu un inversor).

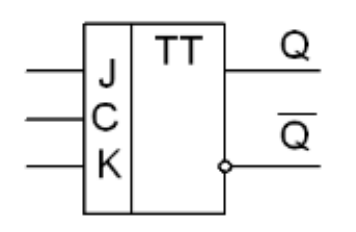


Figura 7. Bistabil JK

**Bistabilul T**

În baza bistabilulele JK și D poate fi obținut bistabilul de tip T, pentru aceasta este necesar să aplicăm la intrările J=K=1.

Bistabilul T este un bistabil de numarare a impulsurilor applicate la intrarea de sincronizare C, este destinat îndeplinirii adunării între informația care se pastrează în bistabil și cea care se transmite la intrarea T a bistabilului. Această particularitate piermite utilizarea acestui bistabil în număratoare.

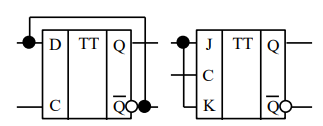


Figura 8. Bistabil T

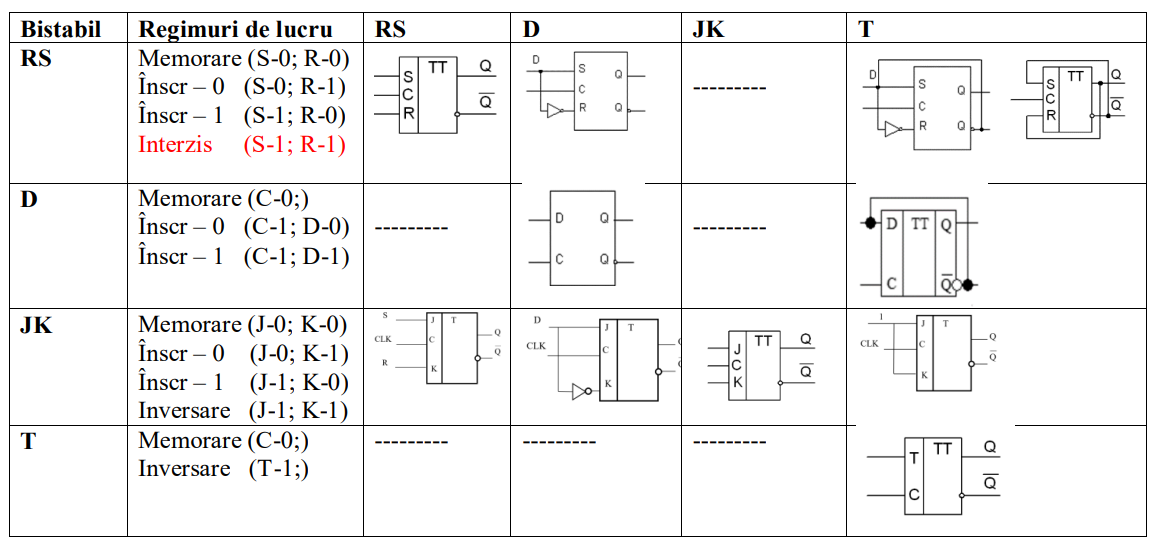


Figura 9. Tipuri de Bistabilul și regimuri de lucru

### **Registrele**

Registrul – este un element funcţional realizat în baza unui circuit logic secvenţial şi este destinat păstrării şi procesării informaţiei în calculator. Componenta de bază a oricărui registru este bistabilul.

Numărul de bistabile ce intră în componența Rg (registrului) este egal cu numărul biţilor cuvîntului de calculator care poate fi păstrat în registru și se numește rangul registrului. Registrele pot fi realizate în baza oricăror tipuri de bistabile în dependenţă de funcţiile pe care le îndeplinesc şi unităţile de calcul unde registrele sunt parte componentă.

Sunt următoarele categorii de registre:

* + Registre cu acces în paralel – adică toţi biţii cuvîntului se înscriu sau se citesc concomitent.
  + Registre cu acces mixt – în aceste registre informaţia se înscrie în paralel şi se citeşte succesiv, sau se înscrie succesiv şi se citeşte în paralel.
  + Registre cu acces succesiv – în aceste registre informaţia se înscrie şi se citeşte pe rând bit cu bit.

Înscrierea şi citirea informaţiei în paralel poate fi efectuată sau în cod direct sau în cod parafazic. În primul caz informaţia care se înscrie în registru se transmite printr-o magistrală de n biţi unde n biţi este numărul rangurilor registrului. În al doilea caz informaţia se transmite printr-o magistrală de 2n biţi concomitent în cod direct şi invers.

Într-un registru informaţia se înscrie în două etape, toate bistabilele registrului se resetează pentru aceasta la intrarea R a tuturor bistabilelor se aplică unu. Prin magistrala de date prin toate rangurile se transmit zerouri, iar la intrările se aplică un impuls. Registrele sunt realizate preponderent în baza bistabilelor D.

Ezistă căteva regimuri de lucru a registrelor:

* Deplasarea dreapta
* Deplasarea stânga
* Încărcarea datelor
* Pastrarea informației

Numărul semnalelor de comandă a registrului se calculează prin formula: log2m=k Unde m – totalitatea regimurilor de funcționare a registrului; iar k – numărul necesar de semnale de comandă;

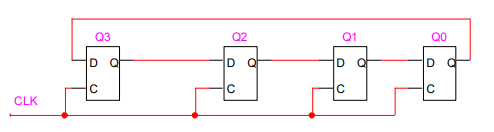


Figura 10. Sinteză registrului pe 4 biți deplasare ciclică dreapta.

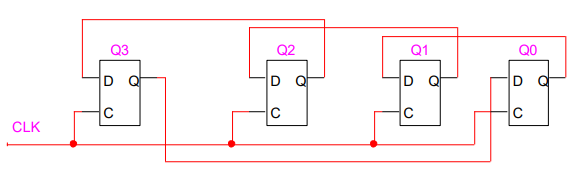


Figura 11. sinteză registrului pe 4 biți deplasare ciclică dreapta.

### **Numărătoarele**

Definiţie:

Numărătoarele (contoarele) – elementele funcţionale segvenţiale destinate numărării impulsurilor aplicate la intrarea lui şi divizarea frecvenţei acestor impulsuri. Componenta de bază a numărătoarelor sunt bistabilele. În componenţa numărătoarelor pot intra toate tipurile de bistabile.

Un numărător are M stări distincte. Tranziţia între stările succesive se produce în urma impulsurilor aplicate la intrarea numărătorului. După aplicarea unui număr de M impulsuri, numărătorul revine în starea iniţială (de exemplu în starea zero). Un astfel de circuit reprezintă un numărător modulo M.

Relaţia dintre modulo M şi numărul de bistabile n, ce intră în componenţa numărătorului este următoarea: n =[log2M]. Dacă numărul n nu este întreg, atunci el se rotungeşte pînă la cel mai apropiat număr întreg mai mare decît cel obţinut.

Modulo – este numărul de stări distincte ale numărătorului. Stare distinctă – combinaţia valorilor stărilor bistabilelor.

Numărătoarele se clasifică după 2 criterii:

* În dependenţă de faptul cum î-şi schimbă starea bistabilele care intră în componenţa numărătorului:
* sincrone
* asincrone

În numărătoarele sincrone bistabilele î-şi schimbă starea concomitent. În cele asincrone - succesiv

* În dependenţă de ordinea numărării:
* directe
* inverse
* reversibile

În cele directe – starea numărătorului se schimbă în ordinea cresccătoare. În inverse – se schimbă descrescătoare. În reversibilă – numărarea se face în ambele direcţii. Acest lucru depinde de regimul numărătorului la momentul dat.

**Numărătoare asincrone**

Specificul numărătoarelor asincrone constă în faptul că ele pot fi realizate sau în baza bistabilelor T sau a altor bistabile, dar care prin conexiuni suplimentare se ajustează tot ca bistabil de tip T .

În numărătoarele directe asincrone ieşirile directe ale bistabilelor se conectează la intrările de sincronizare ale bistabilelor ce se află în poziţia vecină mai semnificativă.

În numărătoarele asincrone inverse conexiunea între bistabile din rangurile vecine se efectuează prin conexiunea ieşirelor inverse la intrările de sincronizare.

În numărătoarele reversibile conexiunea dintre ieşirile bistabilelor şi intrarea de sincronizare se face prin circuite de comutare care în dependenţă de regimul de funcţionare conectează sau ieşirea directă sau ieşirea inversă la intrarea de sincronizare a bistabilului vecin.

**Sinteza numărătoarelor sincrone**

* Sinteza numărătoarelor sincrone directe şi inverse
* Sinteza numărătoarelor sincrone reversibile

Principalul factor care deosebeşte numărătoarele sincrone de cele asincrone constă în faptul că toate bistabilele acestor numărătoare î-şi schimbă starea concomitent şi deci intrările de ceas a tuturor bistabilelor acestor numărătoare se conectează între ele iar la celelalte în dependență de valoarea semnalelor aplicate la intrările bistabilelor.

Se poate observa din analiza succesiunii directe (tabelul 1) că oricare bistabil trebuie să-şi inverseze starea numai atunci cînd toate bistabilele de rang inferior sunt în starea de unu logic.

Pentru un numărător sincron modulo 16 condiţiile de basculare se stabilesc conform relaţiilor: T0=1, T1=Q0, T2=Q0Q1, T3=Q0Q1Q2. Schema numărătorului este prezentată în următoarea figură. În acest numărător este implementată propagarea paralelă a transportului.

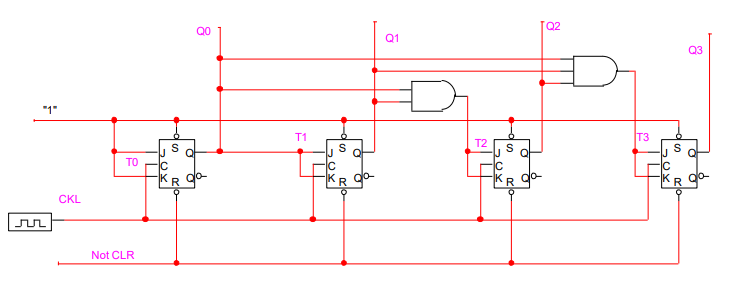


Figura 12. Implementată propagarea paralelă a transportului a Numarătorului.

De aceea sinteza unui numărător sincron modulo M se efectuează în cîteva etape, care sunt următoarele:

* Se determină numărul de bistabile ale numărătorului conform relaţiei n=[log2M];
* Se elaborează tabelul de tranziţie al numărătorului, în care se completează coloanele pentru starea prezentă a numărătorului (momentul t), starea următoare (momentul t+1) şi valorile ce trebuie aplicate la intrările tuturor bistabilelor pentru a asigura tranziţia numărătorului din starea de la momentul t în starea de la momentul t+1.
* Din tabelul de tranziţie al numărătorului se obţin funcţiile de instalare a bistabilelor, care se minimizează;
* În baza funcţiilor minimizate se elaborează circuitele de conexiune a bistabilelor între ele în urma implementării cărora se obţine schema numărătorului.

# **Proectarea convertorului de cod**

## **Minimizarea FDNP și FCNP.**

Efectuarea sintezei unui convertor de cod binar-zecimal FDNP și FCNP.Funcţiile sunt reprezentate în forma disjunctivă normală perfectă şi forma disjunctivă minimală. Pentru forma minimală să se prezinte schema în setul de elemente ŞI-NU și SAU-NU.

Se asamblează şi se reglează schema convertorului de cod binar-zecimal în setul de elemente ŞI-NU și SAU-NU.Pentru circuitele asamblate se determină costul şi timpul de reţinere

|  |  |  |
| --- | --- | --- |
| Varianta | Intrare | Ieșire |
| 27 | 8 6 1 (-4) | 4 3 2 1 |

Tabelul 1. Tabelul de adevăr

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 8 | 6 | 1 | -4 | 4 | 3 | 2 | 1 |
| X1 | X2 | X3 | X4 | Y1 | Y2 | Y3 | Y4 |
| 0(0) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1(2) | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 2(5) | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 3(7) | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 4(9) | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 5(11) | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 6(4) | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 7(6) | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 8(8) | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 9(10) | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 10 | 0 | 0 | 0 | 1 | \* | \* | \* | \* |
| 11 | 0 | 0 | 1 | 1 | \* | \* | \* | \* |
| 12 | 1 | 1 | 0 | 0 | \* | \* | \* | \* |
| 13 | 1 | 1 | 0 | 1 | \* | \* | \* | \* |
| 14 | 1 | 1 | 1 | 0 | \* | \* | \* | \* |
| 15 | 1 | 1 | 1 | 1 | \* | \* | \* | \* |

### **Minimizarea funcțiilor folosint metoda lui Quine-Mc-Cluskey: FDNP**

Tabelul 2. Minimizarea funcției y1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Intrari | | Prima | | A Doua | | Implicatii primare |
| 1 | (1)0001 | 1 | (6.4)01-0 | 1 | (14,12,6,4)-1-0 | (14,12,6,4)-1-0 |
| (4)0100 | (12.4)-100 | (14,12,10,8)1--0 | (14,12,10,8)1--0 |
| (8)1000 | (10,8)10-0 | 2 | (15,14,11,10)1-1- | (15,14,11,10)1-1- |
| 2 | (6)0110 | (12,8)1-00 | (15,14,13,12)11-- |
| (10)1010 | (3,1)00-1 |
| (3)0011 | 2 | (14,6)-110 |
| (12)1100 | (11,10)101- |
| 3 | (11)1011 | (14,10)1-10 |
| (13)1101 | (11,3)-011 |
| (14)1110 | (13,12)110- |
| 4 | (15)1111 | (14,12)11-0 |
|  |  | 3 | (15,11)1-11 |
|  |  | (15,13)11-1 |
|  |  | (15,14)111- |

Tabelul 3. Minimizarea funcției y2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Intrari | | Prima | | A Doua | | Implicatii primare |
| 1 | (8)1000 | 1 | (9,8)100- | 1 | (13,12,9,8)1-0- | (13,12,9,8)1-0- |
|  | (1)0001 | (10,8)10-0 | (14,12,10,8)1--0 | (14,12,10,8)1--0 |
| 2 | (12)1100 | (12,8)1-00 | 2 |  | (14,6)-110 |
| (6)0110 | (9,1)-001 | (15,14,13,12)11-- |
| (9)1001 | (3,1)00-1 |
| (10)1010 | 2 | (14,6)-110 |
| (3)0011 | (13,9)1-01 |
| 3 | (13)1101 | (14,10)1-10 |
| (14)1110 |  |
|  | (13,12)110- |
| 4 | (15)1111 | (14,12)11-0 |
|  |  | 3 |  |
|  |  | (15,13)11-1 |
|  |  | (15,14)111- |

Tabelul 4. Minimizarea funcției y3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Intrari | | Prima | | A Doua | | Implicatii primare |
| 1 | (4)0100 | 1 | (5,4)010- | 1 | (13,12,5,4)-10- | (13,12,5,4)-10- |
|  | (1)0001 | (12,4)-100 | (7,3,5,1)0--1 | (7,3,5,1)0--1 |
| 2 | (5)0101 | (5,1)0-01 | 2 | (15,13,7,5)-1-1 | (14,10)1-10 |
| (10)1010 | (3,1)00-1 | (15,14,13,12)11-- |
| (3)0011 |  |
| (12)1100 | 2 | (7,5)01-1 |
|  | (13,5)-101 |
| 3 | (7)0111 | (14,10)1-10 |
| (13)1101 | (7,3)0-11 |
| (14)1110 | (13,12)110- |
| 4 | (15)1111 | (14,12)11-0 |
|  |  | 3 | (15,7)-111 |
|  |  | (15,13)11-1 |
|  |  | (15,14)111- |

Tabelul 5. Minimizarea funcției y4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Intrari | | Prima | | A Doua | | Implicatii primare |
| 1 | (2)0010 | 1 | (3,2) 001- | 1 | (13,12,9,8)1-0- | (13,12,9,8)1-0- |
| (8)1000 | (9,8)100- | (11,3,9,1)-0-1 | (15,11,7,3)--11 |
| (1)0001 | (12,8)1-00 | 2 | (15,13,11,9)1--1 | (3,2) 001- |
| 2 | (9)1001 |  | (15,14,13,12)11-- |
| (9)1001 | (3,1)00-1 |  | (15,11,7,3)--11 |
| (3)0011 | (9,1)-001 |
| (12)1100 | 2 | (11,9)10-1 |
|  | (13,9)1-01 |
| 3 | (7)0111 | (7,3)0-11 |
| (11)1011 | (11,3)-011 |
| (13)1101 | (13,12)110- |
| (14)1110 | (14,12)11-0 |
| 4 |  | 3 | (15,7)-111 |
| (15)1111 | (15,13)11-1 |
|  | (15,14)111- |
|  | (15,11)1-11 |

În rezultatul minimizării au fost obţinute următoarele funcţii logice:

Luînd în consideraţie conjuncţiile comune, funcţiile y4, y3, y2, y1 pot fi scrise în felul următor:

Unde:

Aducerea funcțiilor la forma elementară *ȘI-NU*

Schema în setul de elemente ŞI-NU:

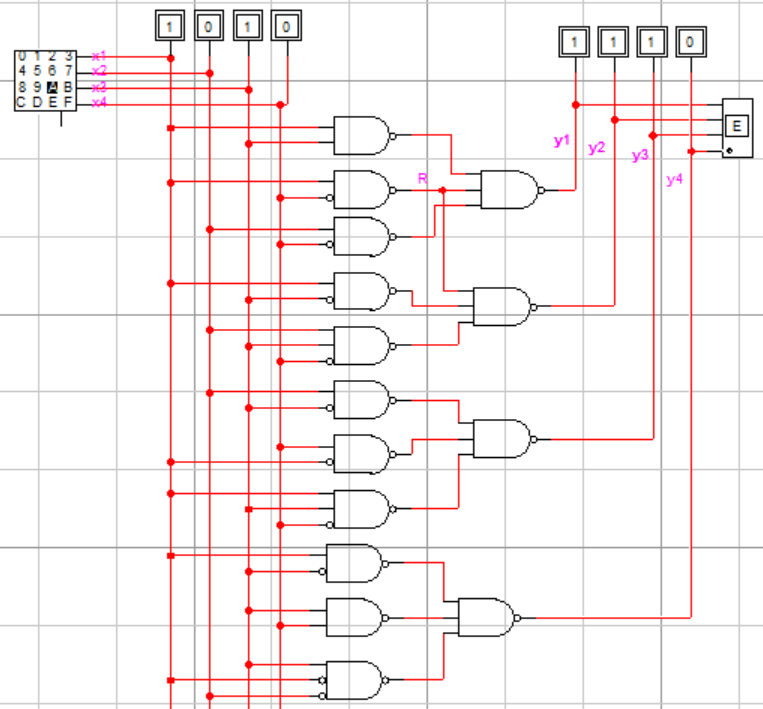


Figura 13. Reprezentarea funcțiilor în Logic Works

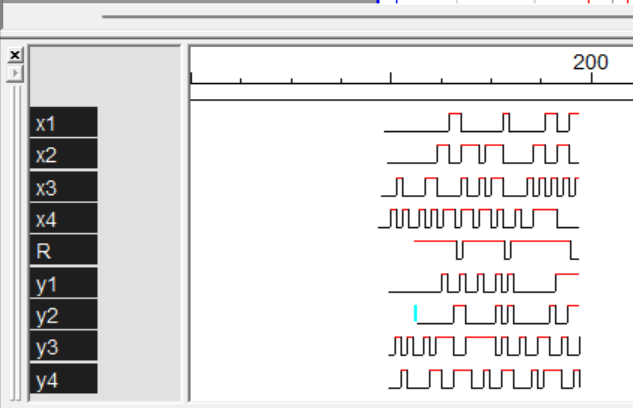


Figura 14. Diagrama de timp

Costul 37

Timpul de retinere a segmentului Td:2

### **Minimizarea funcțiilor folosint metoda lui Karno: FCNP**

Tabelul 6. Minimizarea funcției y1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x1x2  x3x4 | 00 | 01 | 11 | 10 |
| 00 | 0 |  | \* |  |
| 01 | \* | 0 | \* | 0 |
| 11 | \* | 0 | \* |  |
| 10 | 0 |  | \* |  |

Tabelul 7. Minimizarea funcției y2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x1x2  x3x4 | 00 | 01 | 11 | 10 |
| 00 | 0 |  | \* |  |
| 01 | \* | 0 | \* |  |
| 11 | \* | 0 | \* | 0 |
| 10 | 0 |  | \* |  |

Tabelul 8. Minimizarea funcției y3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x1x2  x3x4 | 00 | 01 | 11 | 10 |
| 00 | 0 |  | \* | 0 |
| 01 | \* |  | \* | 0 |
| 11 | \* |  | \* | 0 |
| 10 | 0 | 0 | \* |  |

Tabelul 9. Minimizarea funcției y4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x1x2  x3x4 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | \* |  |
| 01 | \* | 0 | \* |  |
| 11 | \* |  | \* |  |
| 10 |  | 0 | \* | 0 |

În rezultatul minimizării au fost obţinute următoarele funcţii logice:

Luînd în consideraţie conjuncţiile comune, funcţiile y4, y3, y2, y1 pot fi scrise în felul următor:

Unde:

Aducerea funcțiilor la forma elementară *SAU-NU*

Schema în setul de elemente SAU-NU:

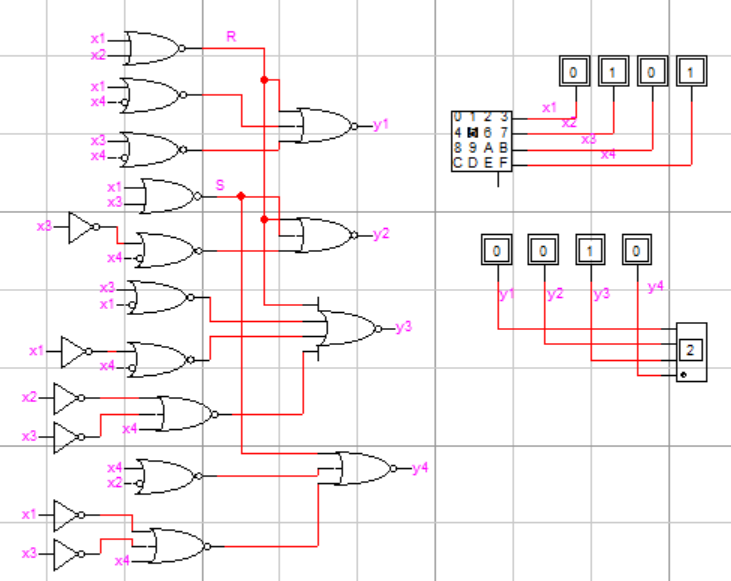


Figura 15. Reprezentarea funcțiilor în Logic Works

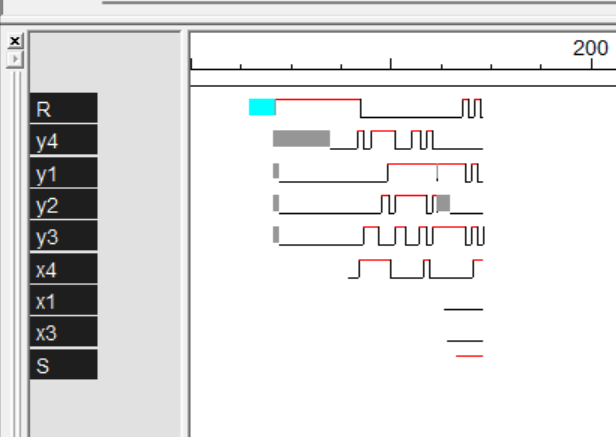


Figura 16. Diagrama de timp

Costul 41

Timpul de retinere a segmentului Td:3

## **Construirea și simularea convertorului, efectuaţi sinteze-i unui binar-zecimal pe 7 segmente**

Tabel.10. Codul 4321

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cifra  Zeci-  mala | codul | | | | DC 7 Segmente | | | | | | |
| 4 | 3 | 2 | 1 |
| X1 | X2 | X3 | X4 | a | b | c | d | e | f | g |
| 0(0) | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1(1) | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2(2) | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 3(3) | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 4(5) | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 5(9) | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 6(10) | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 7(12) | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 8(13) | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 9(14) | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 10 | 0 | 1 | 0 | 0 | \* | \* | \* | \* | \* | \* | \* |
| 11 | 0 | 1 | 1 | 0 | \* | \* | \* | \* | \* | \* | \* |
| 12 | 0 | 1 | 1 | 1 | \* | \* | \* | \* | \* | \* | \* |
| 13 | 1 | 0 | 0 | 0 | \* | \* | \* | \* | \* | \* | \* |
| 14 | 1 | 0 | 1 | 1 | \* | \* | \* | \* | \* | \* | \* |
| 15 | 1 | 1 | 1 | 1 | \* | \* | \* | \* | \* | \* | \* |

Construim diagramele Karnaugh pentru minimizare funcțiilor a-g.

pentru a avem:

1. Tabelul 10.1. Minimizarea funcției a

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x1x2  x3x4 | 00 | 01 | 11 | 10 |
| 00 | 1 | \* | 1 | \* |
| 01 |  |  | 1 | 1 |
| 11 | 1 | \* | \* | \* |
| 10 | 1 | \* | 1 | 1 |

1. Tabelul 10.2. Minimizarea funcției b

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x1x2  x3x4 | 00 | 01 | 11 | 10 |
| 00 | 1 | \* | 1 | \* |
| 01 | 1 | 1 | 1 |  |
| 11 | 1 | \* | \* | \* |
| 10 | 1 | \* | 1 |  |

1. Tabelul 2.3. Minimizarea funcției c

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x1x2  x3x4 | 00 | 01 | 11 | 10 |
| 00 | 1 | \* | 1 | \* |
| 01 | 1 | 1 | 1 | 1 |
| 11 | 1 | \* | \* | \* |
| 10 |  | \* | 1 | 1 |

1. Tabelul 10.4. Minimizarea funcției d

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x1x2  x3x4 | 00 | 01 | 11 | 10 |
| 00 | 1 | \* |  | \* |
| 01 |  |  | 1 | 1 |
| 11 | 1 | \* | \* | \* |
| 10 | 1 | \* | 1 | 1 |

1. Tabelul 10.5. Minimizarea funcției e

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x1x2  x3x4 | 00 | 01 | 11 | 10 |
| 00 | 1 | \* |  | \* |
| 01 |  |  | 1 |  |
| 11 |  | \* | \* | \* |
| 10 | 1 | \* |  | 1 |

1. Tabelul 10.6. Minimizarea funcției f

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x1x2  x3x4 | 00 | 01 | 11 | 10 |
| 00 | 1 | \* |  | \* |
| 01 |  | 1 | 1 | 1 |
| 11 |  | \* | \* | \* |
| 10 |  | \* | 1 | 1 |

1. Tabelul 10.7. Minimizarea funcției g

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x1x2  x3x4 | 00 | 01 | 11 | 10 |
| 00 |  | \* |  | \* |
| 01 |  | 1 | 1 | 1 |
| 11 | 1 | \* | \* | \* |
| 10 | 1 | \* | 1 | 1 |

În rezultatul minimizării au fost obţinute următoarele funcţii logice:

Luînd în consideraţie conjuncţiile comune, funcţiile a, b, c, d, e, f, g pot fi scrise în felul următor:

Unde:

**Aducerea funcțiilor la forma elementară *ȘI-NU:***

Schema în setul de elemente ŞI-NU:

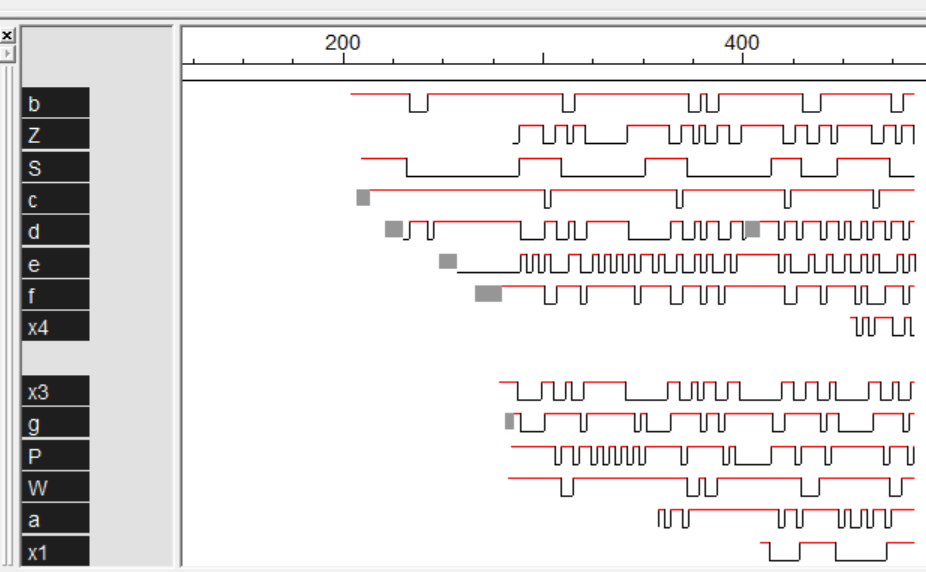


Figura 17. Diagrama de timp

Costul 52

Timpul de retinere a segmentului Td:2

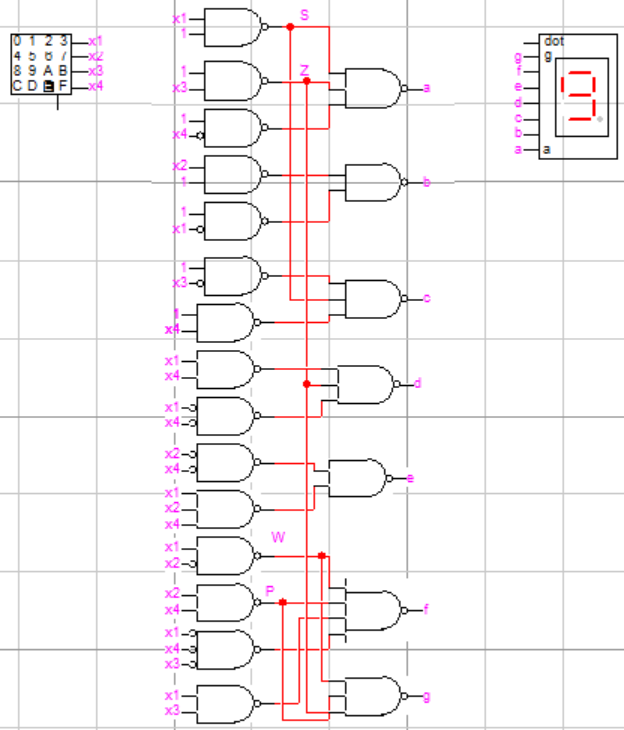


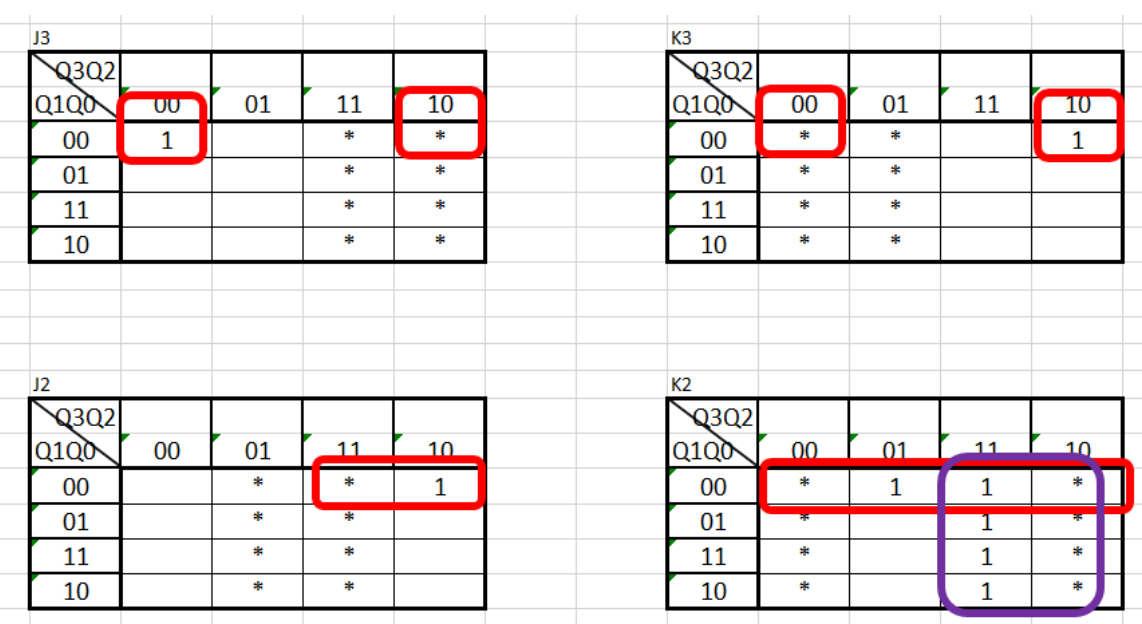
Figura 4. schema unui convectorul de cod binar-zecimal pe 18 segmente.

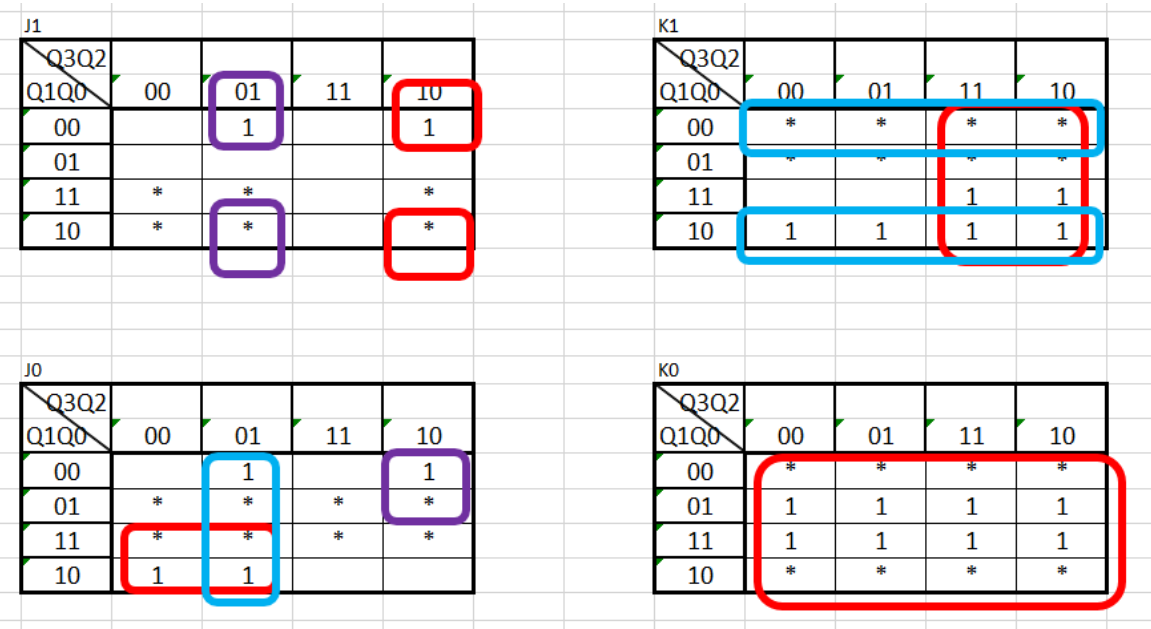
## **Circuit login secvential (Numaroator asincron)**

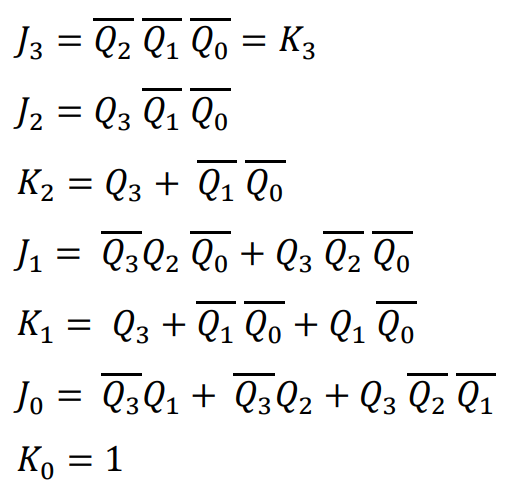
|  |  |  |  |
| --- | --- | --- | --- |
| Nr. Var | Modul de numarare | Tip Bistabil | Tip Numarator |
| 27 | 20 | JK | Invers |

Tabel 11. Numarator asingron

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Intrarile Bistabilelor | | | | Iesirile Bistabilelor | | | | Functiile de instalare a bistabilelor | | | | | | | |
| t | | | | t+1 | | | |
| Q3 | Q2 | Q1 | Q0 | Q3 | Q2 | Q1 | Q0 | J3 | K3 | J2 | K2 | J1 | K1 | J0 | K0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | \* | 0 | \* | 0 | \* | 0 | \* |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | \* | 0 | \* | 0 | \* | \* | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | \* | 0 | \* | \* | 1 | 1 | \* |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | \* | 0 | \* | \* | 0 | \* | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | \* | \* | 1 | 1 | \* | 1 | \* |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | \* | \* | 0 | 0 | \* | \* | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | \* | \* | 0 | \* | 1 | 1 | \* |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | \* | \* | 0 | \* | 0 | \* | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | \* | 1 | 1 | \* | 1 | \* | 1 | \* |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | \* | 0 | 0 | \* | 0 | \* | \* | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | \* | 0 | 0 | \* | \* | 1 | 0 | \* |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | \* | 0 | 0 | \* | \* | 1 | \* | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | \* | 0 | \* | 1 | 0 | \* | 0 | \* |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | \* | 0 | \* | 1 | 0 | \* | \* | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | \* | 0 | \* | 1 | \* | 1 | 0 | \* |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | \* | 0 | \* | 1 | \* | 1 | \* | 1 |







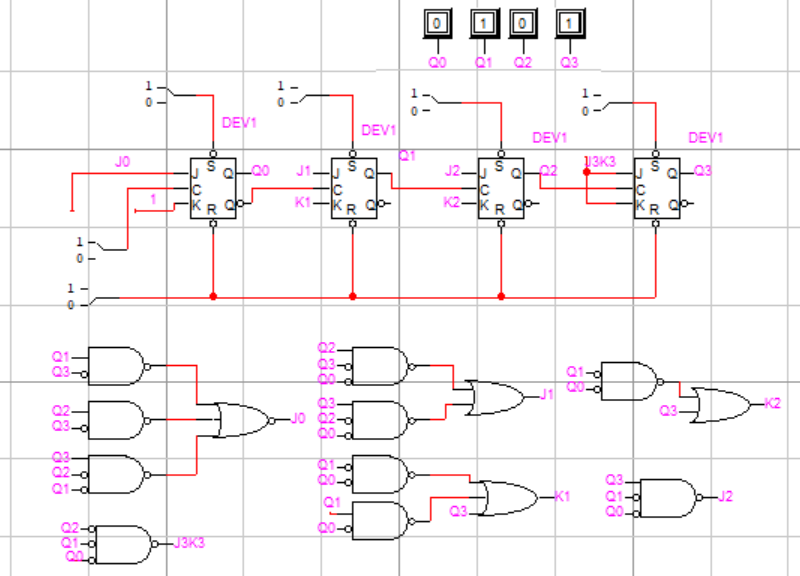


Figura 19. schema unui Numaroator asincron

## **Transformările binare.**

* (4567+27)10->(0001000111110010)2->(10762)8->(11F2)16

(4567+27)10=(4594)10

(4594)10=(0001000111110010)2

4594:2=2297 /0

2297:2=1148 /1

1148:2=574 /0

574:2=287 /0

287:2=143 /1

143:2=71 /1

71:2=35 /1

35:2=17 /1

17:2=8 /1

8:2=4 /0

4:2=2 /0

2:2=1 /0

1:2=0 /1

(0.001.000.111.110.010)2=(10762)8

(0001.0001.1111.0010)2=(11F2)16

* (10110110001100+(27)2)2->()8->()16->()10

27:2=13 /1 13:2=6 /1 6:2=3 /0 3:2=1 /1 1:2=0 /1 (27)10=(11011)2

10110110 001100+

11011

10110110100111

(010.110.110.100.111)2=(26647)8

(0010.1101.1010.0111)2=(2DA7)16

(10110110100111)₂ = (1 × 2¹³) + (0 × 2¹²) + (1 × 2¹¹) + (1 × 2¹⁰) + (0 × 2⁹) + (1 × 2⁸) + (1 × 2⁷) + (0 × 2⁶) + (1 × 2⁵) + (0 × 2⁴) + (0 × 2³) + (1 × 2²) + (1 × 2¹) + (1 × 2⁰) = (11687)₁₀

# **Concluzie:**

Scopul principal este studierea bazelor logice şi aritmetice ale calculatoarelor numerice, însuşirea metodelor de analiză şi sinteză a circuitelor logice combinaţionale şi secvenţiale pentru a le permite studenţilor să analizeze, proiecteze şi implementeze dispozitive numerice.

* Cunoaşterea şi definirea noţiunilor de bază din algebra booleană, necesare pentru studierea aprofundată a metodelor de analiză şi sinteză a dispozitivelor numerice;
* Utilizarea metodelor de  minimizare a funcţiilor logice;
* Cunoaşterea sistemelor de numeraţie utilizate în dispozitivele numerice şi a formelor de reprezentare a numerelor fracţionare şi întregi;
* Utilizarea algoritmilor de bază pentru efectuarea operaţiilor aritmetice binare de adunare, înmulţire şi împărţire în dispozitivele numerice;
* Obţinerea experienţei şi îndemânărilor practice în sinteza circuitelor combinaţionale şi secvenţiale.
* Alegerea criteriilor și metodelor pentru analiza avantajelor și dezavantajelor metodelor și procedeelor aplicate la soluționarea problemelor de calcul numeric.

În concluzie pot spune destul de bine a fost însușite sarcinile propuse și anume:

Am realizat în urma acestui proect transformarile binary/octal/hexazecimal/zecimal, minimizarea funcțiilor prin metoda carno pentru FDNP și FCNP, la fel și crearea shemelor logice în programul LogicWorks 4.0. Ieșirile Convertorului de cod de 7 segmente, realizarea minimizări și schema, în schemă este reprezentat afișarea cifrelor. Și pentru final am realizat implementarea unui numerător asincron, crearea tabelului de adevăr, minimizarea carno și respective chema necesară.

# **Bibliografie:**

<https://pluginfile.php/32954/mod_resource/content/4/T%203.1%20Sisteme%20de%20numeratie%20codurile.pdf>

<https://pluginfile.php/78543/mod_resource/content/4/T%203.2%20Conversia%20numerelor.pdf>

<https://users.utcluj.ro/~baruch/book_ac/AC-Minimiz.pdf>

<https://koaha.org/wiki/Metodo_di_Quine-McCluskey>

<https://pluginfile.php/59777/mod_resource/content/2/T%201.2%20Forme%20de%20reprezentare%20a%20FL%20.pdf>

<https://pluginfile.php/53794/mod_resource/content/2/T%207%20Sinteza%20circuitelor%20secven%C5%A3iale%20%281%29.pdf>

https://pluginfile.php/78546/mod\_resource/content/2/T%207.2%20Numaratoare.pdf

<https://pluginfile.php/53795/mod_resource/content/1/Registre.pdf>